FINDING THE OPTIMUM PARTITIONING FOR MULTI-STANDARD RADIO SYSTEMS

Hans-Martin Bluethgen, Christian Sauer, Matthias Gries, Wolfgang Raab,
Dominik Langen, Alexander Schackow, Manuel Loew, Ulrich Hachmann,
Nico Bruels, Ulrich Ramacher
Infineon Technologies AG, 81609 Munich, Germany,
Hans-Martin.Bluethgen@infineon.com

ABSTRACT

Future wireless communication systems must support multiple radio standards and be capable of executing them concurrently. Consequently, flexibility and programmability will be two of the most important features. However, even a flexible solution has to fulfill the traditional constraints on silicon area and power dissipation. In this paper we address the problem of designing a cost-efficient radio solution starting from the application level. We describe a methodology that drives the design in five phases from application to implementation. We apply this approach in the design of our solution for software-defined radio (SDR) terminals. Here, an optimum partitioning between all components of a multi-standard transceiver system has to be found. The outcome is a heterogeneous platform with a multi-standard baseband circuit and multiple front-ends. The baseband circuit consists of multiple DSP cores and dedicated accelerators and is fully software-programmable.

1. INTRODUCTION

Flexibility is becoming one of the most important design criteria in future terminals for mobile communications and wireless networks. In contrast to today's dual-band single-standard cell phones, future wireless terminals will have to be multi-band, multi-standard and able to do handover between multiple standards and execute them concurrently. With the further evolution from 2G to 4G communication systems more and more standards have to be supported by and integrated into wireless terminals [1]. In addition, increasing design costs, e.g. in terms of manufacturing and testing, require flexible platforms for design reuse.

On the other hand, silicon solutions for radio handsets feature particularly tough constraints on silicon area and power dissipation. Therefore, it is mandatory to carefully design all components of a radio system taking trade-offs between components and HW/SW into account. In this paper we present a multi-standard radio solution for mobile terminals including all components from RF front-end to baseband. Finding the optimum partitioning among the components and exploring HW/SW trade-offs follows a top-down approach for the design of cost-efficient platforms for communication systems. Our main goal is to support the designer with quantitative estimations of the design quality as early as possible during the design flow, where partitioning decisions have most impact on the performance and costs of the platform.

This is why we have developed this approach taking into account the importance of the application domain. One major focus of our methodology is the proper characterization of the domain and implementation of a representative reference application. Using profiling results a fully flexible platform architecture template can be customized and the complexity of the design space significantly reduced to meet costs constraints. Our methodology can be described by the following phases:

1. **Application-domain analysis:** *Identify and define representative and comparable system-level benchmarks and realistic workloads for algorithms and protocols by analyzing the application domain.*

The availability of benchmarks is often an unresolved issue for new application domains. We need to identify and specify representative system-level benchmarks including function, traffic model, and environment specification.

2. **System function modeling:** *Implement a performance indicative system-level reference application that captures essential system functions.*

For performance indicativeness and evaluation of different partitioning and mapping decisions, executable and modular system-level reference applications are required to determine weight and interaction of function kernels. The model must be independent of platform architectures.

3. **Architecture-independent profiling:** *Derive architecture-independent application properties by analyzing and profiling the reference application.*

Our approach particularly emphasizes this step to narrow architectural choices. It is unfeasible to maintain several prototypes of a complex architecture. Before the actual development of the architecture starts we therefore use architecture-independent profiling to derive a good starting point for exploring the platform design space.

4. **Design space exploration:** *Perform a systematic search of the architectural design space based on the reference application and its properties; define and iteratively refine the resulting platform architecture.*

As a starting point, we use a set of embedded standard processors, as well as specialized processing engines. Using the functional reference, we profile the cores to derive a regular multi-processor solution. Such generic solution will be most flexible and reasonably programmable but is likely to fail area and power requirements. Therefore, the profile is used to identify optimization potential and to gradually refine the platform to a virtual prototype with efficient programming environment.

5. **Platform implementation and deployment:** *Implement the platform and provide a code generation framework for the platform to ease the deployment with efficient programming abstractions.*

The virtual prototype serves as executable specification for the actual platform hardware and software implementation and as golden model for verification. In addition, it can make sense to revise an internally used programming abstraction after the exploration if a programmable architecture is released as an open platform to the customer.

In the following sections the five phases of this system-level design approach are explained in detail. The methodology has been applied in case studies from the two domains access platform ([2][3]) and wireless communications. We present the results and experiences we obtained by applying the methodology in our project of a flexible multi-standard radio system for handsets. It has to be noted that due to a lack of appropriate tool support in some phases the methodology cannot be applied as straightforward as it is described, making it necessary to perform some of the steps manually. Section 7 finally closes with concluding remarks.

2. APPLICATION-DOMAIN ANALYSIS

In the first phase of our design flow, the goal is to understand the application domain and derive design requirements and define system environments for further benchmarking. At this early stage of the design flow, essential functions together with their specific parameters need to be derived.

Since we start from application level it is necessary to define reasonable use cases that the final system solution has to support. Figure 1 indicates the numerous wireless communication standards which may have to be integrated into future handsets. Use cases are defined by the set of standards that are supported, and by the combinations of standards that must run concurrently.

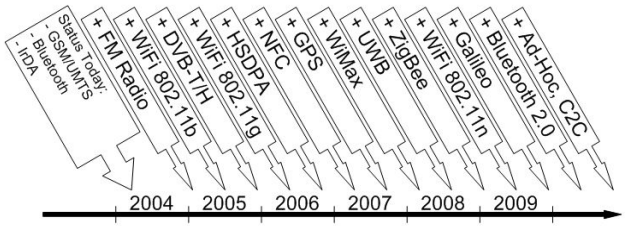

Figure 1: Trend of wireless communication standards (Source: G. Fettweis, TU Dresden).

In general, communication systems consist of analog and digital signal processing sections. The digital section is further partitioned into a digital front-end and the baseband (Figure 2). There is no clear rule for which components of the digital signal processing chain are assigned to the digital front-end or baseband, respectively.

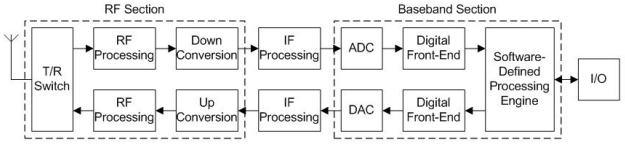

Figure 2: General wireless communication system. The IF processing section is omitted in the case of direct conversion.

For the realization of such a system as a chip set this partitioning usually introduces interfaces between chip boundaries. The current trend is to have an interface on the receive side with a data rate of two times the chip or symbol rate. On the transmit side the interface has a chip or symbol rate interface.

In principle, it is possible to build a fully flexible RF front-end out of programmable or tunable components. These components are wideband antennas, amplifiers, mixers and oscillators, tunable filters, and highly accurate digital synthesizers. However, each of the standards that need to be covered for a multi-standard radio comes with very tight constraints which are difficult to meet even for the single-standard case. Usually, using wideband components means compromising some performance compared to the narrowband solution optimized for the single-standard case. There are currently efforts to combine the analog signal paths of related communication standards into a single flexible RF front-end. Examples of related communication standards are GSM/GPRS/WCDMA. However, it is unlikely that the analog signal paths of different families of communication standards (e.g. wireless LAN and mobile phones) can be combined in the near term. Figure 3 shows the block diagram of a multi-standard radio system featuring multiple RF chips for the different standards. This concept can be easily ex-

tended if more than two signal paths or completely different air interfaces are required.

By comparing standards like WCDMA and WLAN 802.11a it becomes evident, that in the baseband sections functional kernels and topologies are quite diverse.

The partitioning of a system into analog front-end, digital front-end and baseband processor has to take the following issues into account.

1. The interface data rate resulting from a cut between components.
2. The type of interface that is necessary to support this data rate (parallel or serial, single-ended or differential). The type of interface influences pin count and package, and therefore cost.
3. The technologies in which RF and baseband chip are fabricated. This question influences the decision of where to put parts of the digital front-end. It depends on the silicon area they consume if implemented in either of the technologies.

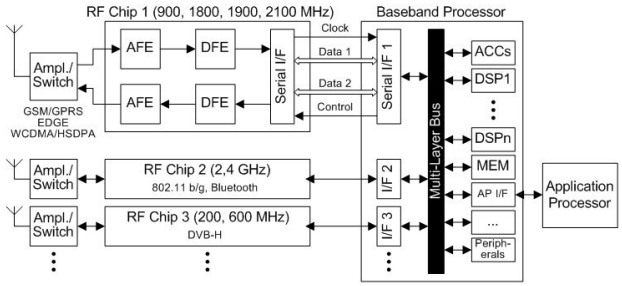

Figure 3: Multi-standard radio system.

Application-domain analysis also includes deriving early estimates on the required processing power. Based on our experience it is necessary to plan with significant headroom in computing power.

The optimization criteria silicon area and power consumption become very important in the design space exploration phase. During analysis it is important to find reasonable ranges that might serve as stop criteria for area/power optimizations.

3. SYSTEM FUNCTION MODELING

In this phase of the design methodology the task is to design a complete model of the system function. A complete system function model in our view has to provide certain features in order to be useful in the subsequent phases of the design methodology.

1. **System function:** *Description of the system function comprised of function kernels, the interconnect between the kernels, and the system control.*
2. **Architectural independency:** *At this phase of the methodology we do not want to impose any restrictions on the implementation of parts of the system, no matter if it is dedicated hardware, reconfigurable structures, or software running on a DSP.*
3. **Simulation:** *The model should be executable and therefore can be used to develop algorithms. It also serves as an executable specification for verification.*
4. **Constraints:** *The model has to contain the entire set of timing, latency, and throughput constraints that is necessary to verify a system implementation or an architecture candidate against the system function model (i.e. the test environment).*

In the communications domain, there are several tools available which are suited mainly for describing the signal processing part of systems, e.g., Matlab/Simulink, CoCentric System Studio, ML Designer, or Ptolemy, to name a few. Currently, none of the tools supports all the features required for a complete system function model as described before. For the modeling of the baseband part of wireless communication systems we use Simulink and extend it with code generation capabilities for parallel processors.

4. ARCHITECTURE-INDEPENDENT PROFILING

After the reference application has been defined, architecture-independent properties of the application are derived by static analysis and/or simulation. The goal of this step is to determine architecture-independent characteristics of the functionality that can help pruning the design space and finding a good starting point in the architecture space. The analysis focuses on features of the reference implementation like data types, primitive operations and sequences thereof, communication between function kernels, or storage requirements. In baseband processing some of these characteristics like primitive operations and sequences thereof can be obtained by profiling the reference model using conventional profiling tools. Others like communication between function kernels have to be estimated from the profiling results for the lack of appropriate tools. In the end we can determine feasible building blocks for the platform architecture as well as lower and upper bounds on the number of computation and communication resources.

5. DESIGN SPACE EXPLORATION

Having found a suitable entry point into the design space the next task is to find the most cost efficient architecture for the platform by exploring the design space. Design parameters used here not only include silicon area and power consumption but also the solution's flexibility and simplicity of the programming model. However, it is not possible to find the optimum solution with respect to so many design parameters in reasonable time. Therefore, we prune the design space by assuming fixed budgets for silicon area and power consumption that have to be agreed upon with the customer.

Finding the most flexible and easy-to-program solution within a specified power and area budget means that we

1. Use the minimum number of general-purpose DSP cores in parallel at the maximum clock frequency and supply voltage so that the power budget is not exceeded,
2. Introduce as few application-specific instructions as possible if the area budget or performance requirements cannot be met with 1.,
3. Use dedicated accelerators for algorithms with high computational complexity if the area budget or performance requirements cannot be met with 2.

With the entry point into the architecture design space chosen as explained above, our estimations on power consumption resulted in an architecture consisting of a cluster of four single-instruction multiple-data (SIMD) DSP cores (Figure 4). This kind of DSP core is particularly suited for the computationally complex algorithms in communication systems [4][5]. Each SIMD core contains four processing elements and operates with a clock frequency of 300 MHz.

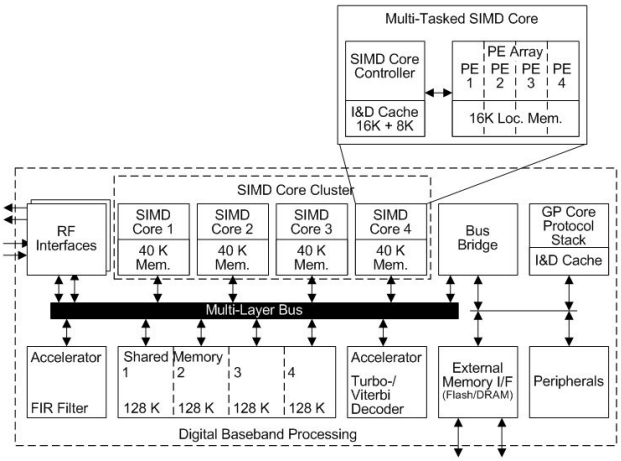

Figure 4: Baseband platform.

Iterative refinement led to

- Specialized instructions for saturating operations and finite-field arithmetic
- Long-instruction word (LIW) features for performing arithmetic operations and memory accesses in parallel
- Multiple task contexts for exploiting concurrency as reflected by the system function model
- Four-stage execution pipeline used in conjunction with the multiple task contexts to relax the timing requirements for the memories, reduce the memory's supply voltage and thereby the power consumption.

The cluster of SIMD cores is accompanied by dedicated programmable processors for channel encoding and decoding as well as filtering operations. These dedicated processors account for almost half of the total processing power of the entire SDR platform [6]. In addition, there is an ARM processor for the execution of the protocol stacks.

We have developed a virtual prototype of the entire platform based on SystemC [7]. The virtual prototype is a cycle- and bit-accurate software-based simulator of the SDR platform. The virtual prototype contains models of all processors, accelerators, busses, memories, and peripherals which will be available in the real hardware. Therefore the same software can be run on the virtual prototype as on the real hardware. Although a virtual prototype need not necessarily be cycle accurate and since the cycle accuracy reduces the simulation speed compared to a transaction accurate model, a cycle accurate virtual prototype offers several advantages over a transaction accurate model. In a transaction accurate model each interaction with its environment, e.g. the transfer of data, is modeled, but basically no timing information for a transaction is generated. However, in some cases it is possible to give an estimation of the duration of a transaction, e.g. the duration of a memory access in a single processor system. In multi-processor systems this is not possible in general in an efficient and accurate way, as the duration of a transaction cannot be estimated locally in one module but has to be derived from the state of the entire system. There may also exist feedback loops, e.g. the higher the utilization of a bus arbiter is, the higher is the probability that a module does not get the bus granted and it has to repeat its request. Consequently, the utilization of the bus arbiter increases further. Cycle-accurate models interact with their environment with the same timing as real hardware. Hence, the duration of a transaction is given implicitly by the model in an absolutly accurate way. Thus we can use our virtual prototype for the SDR application domain to check if our architecture-software combination meets the real-time requirements. Furthermore, cycle-accurate models help to specify modules in detail before writing code on register transfer level (RTL) in a hardware description language (HDL) and they offer the possibility to use the virtual prototype to verify the HDL descriptions.

As described above we are focusing on (heterogeneous) multi-processor platforms. For the SDR prototype, we currently deploy three different types of processors in order to meet the real-time requirements. The first type is a general-purpose core, currently an ARM926 (precompiled SystemC model). The second type is a DSP core with single-instruction multiple-data (SIMD) extensions. This processor is highly configurable, e.g. the number of functional units within the processor can be adapted to the level of data parallelism of the application. The third group of processing units is hardware accelerators for compute-intensive tasks, such as FIR filtering. The accelerators have been modeled completely in SystemC like the rest of the virtual prototype including busses, memories, and peripherals.

The hardware accelerator for channel decoding is a programmable core. 32 parallel add-compare-select units (ACSUs) are available for fast Turbo and Viterbi trellis operations. A trace-back unit for Viterbi and hardware support for

the UMTS Turbo internal interleaver are implemented. A programmable general purpose part is used for parameter preparation, data path configuration, data transfer over the system bus and communication with the system. The decoding operations are performed by a special trellis data path that is controlled by the general purpose part. The special trellis data path is flexible such that Turbo and Viterbi algorithms for various communication standards can be processed. Local memory is available in order to reduce the number of bus transfers and to speed up the decoding process.

The SystemC model is bit-true and cycle-accurate. This means that all relevant signals are modeled with the appropriate bit widths and all pipeline stages are present, i.e. the micro architecture has been modeled in SystemC and the refinement in VHDL must be done only in non-critical places. The SystemC model can be directly translated into VHDL and it serves as a reference for simulation: traces of arbitrary internal signals of the SystemC model can used as stimuli or reference pattern in a VDHL simulation. This eases the verification of the corresponding parts in the VHDL model.

Cycle-accurate modeling of the instruction pipeline of the general purpose part, the six-stage pipeline of the special trellis data path and the access cycles to the local memory provides the exact number of cycles necessary for a decoding process, e.g. the pipeline stalls due to taken branches are included in the model of the instruction pipeline. Since throughput is crucial, the cycle-accurate model enables the verification.

The SDR filter accelerator is a configurable coprocessor for multi-threaded FIR filtering on complex input data. It can locally store up to four different filter contexts at the same time. The datapath can be configured to support different FIR filter modes. Four independent I/O processes for external memory access are running concurrently.

The SystemC SDR filter accelerator model consists of a main controller, a memory interface controller, a datapath and datapath controller, local memories for storing input samples, filter coefficients and output samples, a slave interface, two task FIFOs, and a set of local registers.

In the SystemC model of the filter accelerator a high-level description of both the filter datapath and the datapath controller has been implemented for simulation performance reasons. The datapath latency is modeled by the use of a configurable counter.

The SDR virtual prototype provides several tools for analyzing the soft- and hardware. First, the general purpose cores can be debugged and profiled by using a graphical debugger. Second, internal state, instructions, and data transfers of all modules can be logged. Third, statistical data can be gathered on the utilization of the shared modules like busses and memories. Furthermore, it is possible to track the execution status of threads on the general purpose processors without instrumenting the software. The results can be displayed graphically. An example is shown in Figure 5. An 802.11b implementation is running on the parallel DSP cluster. The traces show single threads running, blocking, and synchronizing.

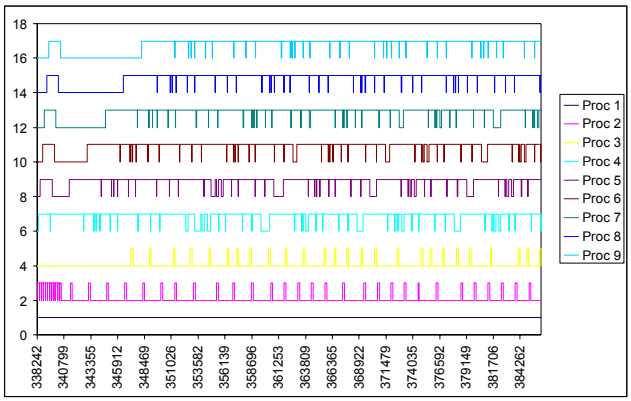

Figure 5: Traces of parallel WLAN 802.11b implementation.

Having the virtual prototype and analysis tools leads to fast and very accurate analyses of the performance of an application. Together with the configurability of the virtual prototype the architecture can be optimized very efficiently.

6. PLATFORM IMPLEMENTATION AND DEPLOYMENT

The virtual prototype serves as an executable specification for the actual platform hardware and software implementation and as a golden model for their verification. As the needed functionality and timing requirements for each module have been accurately analyzed before, hardware designers are relieved from redesigning modules repeatedly due to flaws in the specification. Depending on the level of detail in the virtual prototype, the translation of complete modules to HDL can be straightforward, e.g. if an accelerator is modeled with all pipeline stages and precise widths of all data paths. In this case a lot of iterations in the architecture refinement can be avoided. Another main advantage of having a detailed virtual prototype is that the amount of written specification can be reduced significantly, in our experience from a few thousand to a few hundred pages. This reduces the probability of errors introduced in the design process just by simple misunderstandings.

A very cumbersome task in the chip design process is always the setup of appropriate test benches. Here, for the verification of the virtual prototype test benches with cycle-accurate stimuli have already been written. These test benches can be reused during the platform implementation almost unchanged.

A successful deployment of a programmable platform can only be achieved if the programmer/user is relieved

from understanding the architecture in full detail. An approach is a layer of firmware that provides sufficient abstraction for the customer in an API/library based manner [8]. Ideally, the vendor who understands his platform in full detail provides this layer. This way, only the firmware programmer is faced with the problem of coordinating the execution of software portions on different processing elements manually that must be optimized individually on assembly level – a tedious burden that frequently leads to suboptimal usage of hardware resources. This task would highly benefit from tools that are able to generate at least the part of the software coordinating the parallel execution on different processing elements, ideally from the system function model [9].

7. CONCLUSION

Within the next few years we will see a transition from dual-band single-standard to multi-band multi-standard terminals. Our system-level design methodology for flexible yet cost-efficient communication platforms is based on five phases that emphasizes the role of systematic domain analysis and system function modeling to ease system-level decisions, such as the partitioning of functionality onto platform resources. Starting from system level, a flexible platform architecture is developed and iteratively refined going down to circuit level. This process is application-driven, and relies on the use of virtual prototypes for analyzing and optimizing the architecture using representative reference applications. This virtual prototype is used again in the final deployment phase to facilitate development and verification of RTL code. In a case study the development of a multi-standard radio system demonstrated the application of different domain-specific tools throughout the phases. Tools had to be developed and extended to support the refinement process. The final goal of our effort is to automate at least each of the single phases if not the entire approach.

Currently, a prototype chip of the baseband platform for terminals is being designed in a 90-nm CMOS technology. A multi-standard demonstrator with this chip running WCDMA and 802.11b will be set up by the end of March 2006.

8. ACKNOWLEDGEMENT

The authors wish to thank C. Grassmann, M. Richter, M. Sauermann, and A. Troya for their work and contributions to this project.

9. REFERENCES

[1] U. Ramacher, “Next Generation Embedded Communication Systems: Reconfigurability, Flexibility and Programmability”, Intel Corp. Hillsboro/Oregon, On Chip Reconfigurable Computing and Communications Workshop, May 2003.

[2] C. Sauer, M. Gries, S. Sonntag, “Modular Reference Implementation of an IP-DSLAM”, 10th IEEE Symposium on Computers and Communications (ISCC), June 2005.

[3] C. Sauer, M. Gries, S. Sonntag, “Modular Domain-Specific Implementation and Exploration Framework for Embedded Software Platforms”, 42nd Design Automation Conference (DAC), June 2005.

[4] J.-P. Giacalone, “Trends in Programmable DSP Architecture for new Generation Wireless Modems”, European Solid-State Circuits Conference, Lisbon, September 2003.

[5] G. Fettweis; M. Bolle; J. Kneip; M. Weiss, “OnDSP: A New Architecture for Wireless LAN Applications”, Embedded Processor Forum, San Jose, May 2002.

[6] H.-M. Bluethgen, C. Grassmann, W. Raab, U. Ramacher, J. Hausner, “A Programmable Baseband Platform for Software-Defined Radio”, SDR Technical Conference, Phoenix, USA, November, 2004.

[7] Functional Specification for SystemC 2.0, Update for SystemC 2.0.1, Version 2.0-Q, http://www.systemc.org, April 2002.

[8] C. Grassmann, M. Sauermann, H.-M. Bluethgen, U. Ramacher, "System-Level Hardware Abstraction for Software-Defined Radios", SDR Technical Conference, Phoenix, USA, November, 2004.

[9] R. Hossain, M. Wesseling, C. Leopold, “Virtual Radio Engine: A Programming Concept for Separation of Application Specifications and Hardware Architectures”, 14th IST Mobile & Wireless Communication Summit, Dresden, June 2005.